(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 June 2001 (28.06.2001)

PCT

(10) International Publication Number
WO 01/46819 A2

(51) International Patent Classification[7]: G06F 13/10, H04N 5/00

(21) International Application Number: PCT/US00/42614

(22) International Filing Date: 6 December 2000 (06.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/468,047 21 December 1999 (21.12.1999) US

(71) Applicant *(for all designated States except US)*: GENERAL INSTRUMENT CORPORATION [US/US]; 101 Tournament Drive, Horsham, PA 19044 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: GAZDA, Robert [US/US]; 42 Le Forge Court, Wayne, PA 19087 (US). DEL SORDO, Chris [US/US]; 229 Heatherfield Drive, Souderton, PA 18964 (US). PREZUHY, David, A. [US/US]; 3398 Pin Oak Lane, Chalfont, PA 18914 (US). BIRNBAUM, Jack, M. [US/US]; 559 Nicole Drive, Southampton, PA 18966 (US).

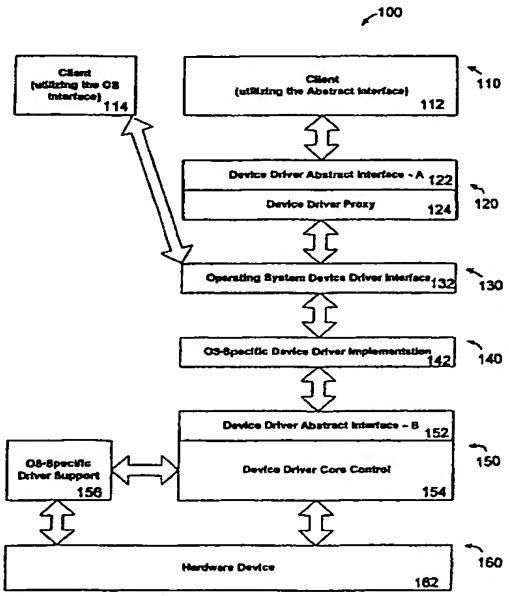
(74) Agent: LIPSITZ, Barry, B.; Law Offices of Barry R. Lipsitz, 755 Main Street, Building No.8, Monroe, CT 06468 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian

[Continued on next page]

(54) Title: ABSTRACT DEVICE DRIVER MODEL FOR THE PORTABILITY OF DEVICE DRIVERS ACROSS DIFFERENT OPERATING SYSTEM PLATFORMS



(57) Abstract: An abstract interface model for device drivers in a set-top terminal, such as for use in a cable or satellite television subscriber network. In a layered software architecture, a device driver abstract interface model allows communication between a client (112, 212) and the device driver (154, 254) regardless of the operating system (OS) under which the device driver operates. In a first embodiment, the architecture includes a dedicated (OS-specific) device driver interface (132), and a proxy uses a first abstract interface (122) to convert interface service requests from a client (112) into OS-specific calls to the device driver (154). For a client that directly accesses a dedicated (OS-specific) device driver interface (132), a second abstract interface (152) is used to convert the interface service requests into OS-specific calls to the device driver (154). Direct access to the OS-specific device driver interface (132) is therefore maintained.

WO 01/46819 A2

patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

ABSTRACT DEVICE DRIVER MODEL FOR THE PORTABILITY OF DEVICE DRIVERS ACROSS DIFFERENT OPERATING SYSTEM PLATFORMS

BACKGROUND OF THE INVENTION

The present invention relates to the field of digital set-top software/firmware for use, e.g., in a cable or satellite television subscriber network. In particular, an abstract interface model is provided for device drivers in a set-top terminal.

The recent advent of digital set-top terminals has spurred the growth of subscriber television networks, such as cable/satellite television networks. Such terminals can support increased levels of programming services and a variety of software-based applications and functions, such as an electronic program guide, stock or weather banners, shop and bank at home services, games, and the like. Moreover, this trend is expected to continue with the convergence of telephone, television and computer networks, and the rise of in-home computer networks.

Each terminal includes different hardware devices, for example, tuners, demodulators, MPEG-2 Decoders (e.g., audio, video, and data), video encoders, audio mixers, and so forth, which are controlled by respective drivers.

Moreover, each terminal uses an operating system (OS) platform for controlling the functions of the terminal. The OS has a hierarchical structure wherein

functions are separated according to their level of abstraction. At the bottom of the structure, the least abstract level or layer is the hardware device level. The next higher level is the device driver level. At the top of the structure, the most abstract level is typically the client level. Additionally, each level manages a set of objects, which can be hardware or software objects, and defines operations that can be carried out on the objects.

However, conventionally it is necessary for a device driver to be implemented separately for each operating system platform, e.g., for different set-top manufacturers, or different set-top models from the same manufacturer. Moreover, a single hardware platform may be required to support different OS platforms. It is even possible for different terminals in the same network to have different operating systems.

Generally, continual operating system changes in set-top terminals is a result of improvements, cost reductions, new components, and second source manufacturers. This is problematic since, even though each network generally uses a single OS platform, the terminal manufacturer must create different device drivers for the different operating systems in different networks. The device driver for the OS of one network cannot be easily migrated to another OS in another network.

The need to develop and provide updated device driver software/firmware to the terminals leads to additional expense for the terminal manufacturer and

network provider.

Specifically, device drivers for a particular OS are tightly bundled to the OS, which means each driver contains OS-specific details throughout the driver's implementation, so a full development cycle is required to develop drivers for each OS platform.

These problem are compounded by the continual upgrading and replacement of terminals in networks as technology advances.

Accordingly, it would be desirable to provide a device driver abstract interface model that allows the development of a single set of device driver code, and device user code that will compile and execute under different set-top operating systems. The interface should allow a device driver to be implemented only once while being usable under several operating systems and set-top platforms. One such suitable platform is the DCT5000 series terminals, manufactured by General Instrument Corporation, the assignee hereof.

The interface should allow two-way communication between the device driver and a client.

The interface should ensure that the hardware device appears the same to the terminal's client (e.g., whatever application or other entity is communicating with the device) regardless of the OS under which the hardware device operates. Similarly, the client should appear the same to the actual device driver regardless of the operating system.

The interface should allow the sharing of architecture and code across both OS platforms and hardware platforms.

The interface should be usable with operating systems that have, or do not have, a dedicated (operating system-specific) device driver interface.

The interface should be implementable using object-oriented or procedure-oriented programming languages.

The present invention provides a device driver abstract interface model having the above and other advantages.

SUMMARY OF THE INVENTION

The present invention relates to an abstract interface model for device drivers in a set-top terminal.

The invention allows the development of driver core logic only once. The core still needs to be compiled separately for each OS platform.

In a particular embodiment, a method is provided for allowing communication between a client and a device driver in a terminal, where the terminal has an associated specific operating system under which the device driver operates. The method includes the step of providing a first abstract interface for converting a service request from the client that has a format that is operating system-independent into a call to the device driver that has a format that is compatible with the specific operating system.

In particular, the first abstract interface is adapted to provide the call to the device driver in any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems.

When a dedicated (OS-specific) device driver interface is present, the device driver interface is adapted to receive a second call directly from another client that has a format that is compatible with the specific operating system. The second call does not pass through the first abstract interface. Moreover, the device driver interface provides the second call to the device driver via a second abstract interface.

A related method for allowing communication between a client and a device driver in a terminal, where the terminal has an associated specific operating system under which the device driver operates, includes the step of providing a first abstract interface for converting a message from the device driver that is in the format of the specific operating system to a corresponding message in an operating system-independent format for use by the client.

The first abstract interface is adapted to convert the message from any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems into the operating system-independent format for use by the client.

Corresponding apparatuses are also presented.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a device driver abstract interface model for use with an operating system that has a dedicated device driver interface in accordance with the present invention.

FIG. 2 illustrates a device driver abstract interface model for use with an operating system that does not have a dedicated device driver interface in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to an abstract interface model for device drivers in a set-top terminal.

Known object-oriented design techniques provide a method for abstractly creating objects. Mainly, the designer can create an object that provides an abstract interface without directly knowing the details of the instantiation of the object. This occurs dynamically, at the run time of the software.

The invention extends or modifies this idea as follows. Only a single driver interface is created for all operating systems. This is done at the time the source code is compiled, and only happens once (statically).

Accordingly, when a new OS platform is developed, at most, a proxy, an OS-specific driver, and the OS-specific driver support require development. This typically represents only about 10% or less of the driver development cycle, thereby resulting in significant cost and time savings.

Specifically, all of the device driver logic is located or implemented in the device driver core control. This module or sub-system resides immediately above the hardware device level, and communicates with the hardware device either directly, or with the assistance of the OS-specific driver support sub-system. Note that the core control includes the majority of all the code in this architecture.

The device driver core control implements the logic defined in the device driver abstract interface. This interface defines what services the driver must provide. The core control includes all code required to provide those services.

FIG. 1 illustrates a device driver abstract interface model for use with an OS that has a dedicated device driver interface in accordance with the present invention. The dedicated device driver interface 130 may be, e.g., Windows CE from Microsoft(tm).

A layered device driver model 100 includes a number of levels 110, 120, 130, 140, 150 and 160. Typically, the OS requires just a driver implemented to its interface, in one level. A bottom level 160, which is the least abstract level, includes the hardware device itself 162 in the set-top terminal.

The next abstract level 150 includes a second device driver abstract interface 152, a device driver core control 154, and an OS-specific driver support 156. The next abstract level 140 includes an OS-specific device driver implementation 142. The next abstract level 130 includes an OS device driver interface 132.

The next abstract level 120 includes the first device driver abstract interface 122, and a device driver proxy 124.

The next abstract level 110, which is the highest level, includes a client 112 that is using the device driver abstract interface 122, and a client 114 that is using the OS device driver interface 132.

Note that while only one device is shown, the invention is suitable for use with any number of devices in a set-top.

In the architecture of FIG. 1, the OS-specific device driver 142 controls or drives the device driver core control 154 using the second device driver abstract interface 152. In particular, this level 140 with the OS-specific device driver 142 converts or transforms OS-specific device driver calls (tied specifically to an OS) into abstract device driver calls that are defined by the device driver abstract interface 152.

The OS itself invokes services within the OS-specific device driver 142 through the operating system's defined driver interface 132. The OS may be Microsoft(tm) Windows CE, for example. This OS invokes calls having the format "xxx_IOCTL" within a Windows CE Device Driver, where "xxx" represents the Windows CE names of the device.

Regarding driver calls, Windows CE represents an example where a proxy would invoke a "DeviceIoControl" call to passing the Win CE name of the device.

The device driver proxy 124 represents the device driver 154 to the client 112 using the abstract interface 122, which is functionally the same as the device driver abstract interface 152 that is implemented by the device driver core control 154. These clients can include, e.g., middleware and applications. Hence, it appears as if the client is interfacing directly to the driver core itself. Moreover, the proxy 124 converts or transforms

interface service requests from the client 112 into their associated OS-specific system device.

Additionally, the abstract interfaces 122, 152 handle data that is communicated from the driver to the client, which can be synchronous or asynchronous. With synchronous communication, the driver 154 only provides data to the client 110, 114 when asked. The data is transferred across the OS driver boundary as described in connection with the client to driver communication.

Asynchronous communication occurs when a driver wishes to send a client data (e.g., a message) without the client asking for it. Note that the driver can only inform the client that it has data, and the client must then synchronously retrieve the data. This is summarized by the following steps:

1. Driver wishes to transfer data to the client.
2. The driver signals the client asynchronously through an event.
3. Client issues a synchronous operation to retrieve or read the data.

To summarize, the device driver proxy 124 converts abstract interface requests from a client 110 into OS-specific device driver system calls. The OS, in turn, invokes OS-specific driver entry points. The OS-specific driver 142 converts the request back into a driver abstract interface request, which is the same interface call that the client originally performed. This request is executed in the device driver core 154.

Finally, external clients can chose two interface paths when accessing drivers. First, the abstract interface 122, 152 can be used in the manner the

internal clients (e.g., clients developed by the set-top manufacturer) use it, such as the client 112 is doing in FIG. 1. Second, the OS driver interface can be directly used, as the client 114 is doing in FIG 1.

An external client is external to the set-top manufacturer. External clients include any entity that needs to use the manufacturer's driver including, e.g., a middleware developer such as Liberate Technologies(tm), San Carlos, California, USA, or an OS developer such as Microsoft.

FIG. 2 illustrates a device driver abstract interface model for use with an operating system that does not have a dedicated device driver interface in accordance with the present invention.

The hierarchical model 200 includes levels 210, 250 and 260. The bottom level 260 includes the hardware device 262. The next abstract level 250 includes a device driver abstract interface 252, a device driver core control 254, and an OS-specific driver support 256.

The next level 210 includes the client 212 that is using the abstract interface 252.

In particular, FIG. 2 refers to a set-top system containing an OS without a device driver interface. An example of such an OS is VRTX from Mentor Graphics Corporation, Wilsonville, Oregon, USA.

This layered architecture compresses (e.g., fewer layers are required) when the OS does not require the use of a dedicated OS-specific interface. Here, the client 212, again, views the device driver 254 using the device driver abstract interface 252. But, the

client directly connects to the driver core 254. Only the abstract interface 252 is available under the OS without dedicated driver interfaces.

Similarly, the device driver 254 views the client via the interface 252 to allow two-way communication.

Accordingly, it can be seen that the present invention provides an abstract interface model for device drivers in a set-top terminal.

The benefits of the invention are as follows. To clients using the abstract interface, the device driver always appears the same. The client does not need to be redeveloped for each operating system. Also, the client cannot tell or see the difference between the two very different OS implementations (see FIGs 1 and 2). The driver appears the same. A key point is that the device drivers support an abstract interface allowing clients to be developed once, and utilized across operating systems.

Additionally, to the driver core control, the client always appears the same. The core logic cannot tell the difference between clients using the abstract interface/proxies, clients using the OS driver interface without the abstract interface/proxies, or clients directly accessing the core, as in FIG. 2.

The driver core control sub-system is implemented in a OS-independent manner. Typically, this sub-system requires a great amount of effort to develop (e.g., up to 90% of all driver development man-hours). However, since the driver core control sub-system of the invention only needs to be developed once, and can then be utilized across different OS platforms, the abstract

device driver model can greatly improve the time to market for new set-top products.

Moreover, the invention allows the use of alternative device driver user interfaces.

Although the invention has been described in connection with various specific embodiments, those skilled in the art will appreciate that numerous adaptations and modifications may be made thereto without departing from the spirit and scope of the invention as set forth in the claims.

For example, while the invention was discussed in connection with a cable or satellite television broadband communication networks, it will be appreciated that other networks such as local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), internets, intranets, and the Internet, or combinations thereof, may be used.

Moreover, the invention may be implemented using any known hardware, firmware and/or software techniques.

What is claimed is:

1. A method for allowing communication between a client and a device driver in a terminal, said terminal having an associated specific operating system under which the device driver operates, comprising the step of:

providing a first abstract interface for converting a service request from the client that has a format that is operating system-independent into a call to the device driver that has a format that is compatible with the specific operating system.

2. The method of claim 1, comprising the further step of:

using the first abstract interface for converting a message from the device driver that is in the format of the specific operating system to a corresponding message in the operating system-independent format for use by the client.

3. The method of claim 1, wherein:

the terminal is a subscriber television terminal in a network.

4. The method of claim 1, wherein:

the terminal is in a broadband communication network.

5. The method of claim 1, wherein:

the first abstract interface is adapted to provide the call to the device driver in any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems.

6. The method of claim 1, comprising the further step of:

providing the call to the device driver via a device driver interface that operates according to the specific operating system.

7. The method of claim 6, wherein:

the device driver interface is adapted to receive a second call directly from another client that has a format that is compatible with the specific operating system; and

the device driver interface provides the second call to the device driver via a second abstract interface.

8. The method of claim 6, wherein:

the first abstract interface is adapted to provide the call to the device driver via the device driver interface in any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems.

9. The method of claim 6, comprising the further step of:

providing a second abstract interface for converting a message from the device driver that is in the compatible format to a corresponding message in the operating system-independent format for use by the client.

10. The method of claim 9, comprising the further step of:

providing the message in the operating system-independent format for use by the client via the first abstract interface.

11. The method of claim 9, wherein:

the second abstract interface is adapted to convert the message to the client from any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems to the client's operating system-independent format.

12. An apparatus for allowing communication between a client and a device driver in a terminal, said terminal having an associated specific operating system under which the device driver operates, comprising:

means for providing a first abstract interface for converting a service request from the client that has a format that is operating system-independent into a call

to the device driver that has a format that is compatible with the specific operating system.

13. A method for allowing communication between a client and a device driver in a terminal, said terminal having an associated specific operating system under which the device driver operates, comprising the step of:

providing a first abstract interface for converting a message from the device driver that is in the format of the specific operating system to a corresponding message in an operating system-independent format for use by the client.

14. The method of claim 13, wherein:

the first abstract interface is adapted to convert the message from any one of a plurality of different formats that are compatible with a corresponding plurality of different operating systems into the operating system-independent format for use by the client.

15. An apparatus for allowing communication between a client and a device driver in a terminal, said terminal having an associated specific operating system under which the device driver operates, comprising:

means for providing a first abstract interface for converting a message from the device driver that is in the format of the specific operating system to a

corresponding message in an operating system-independent format for use by the client.

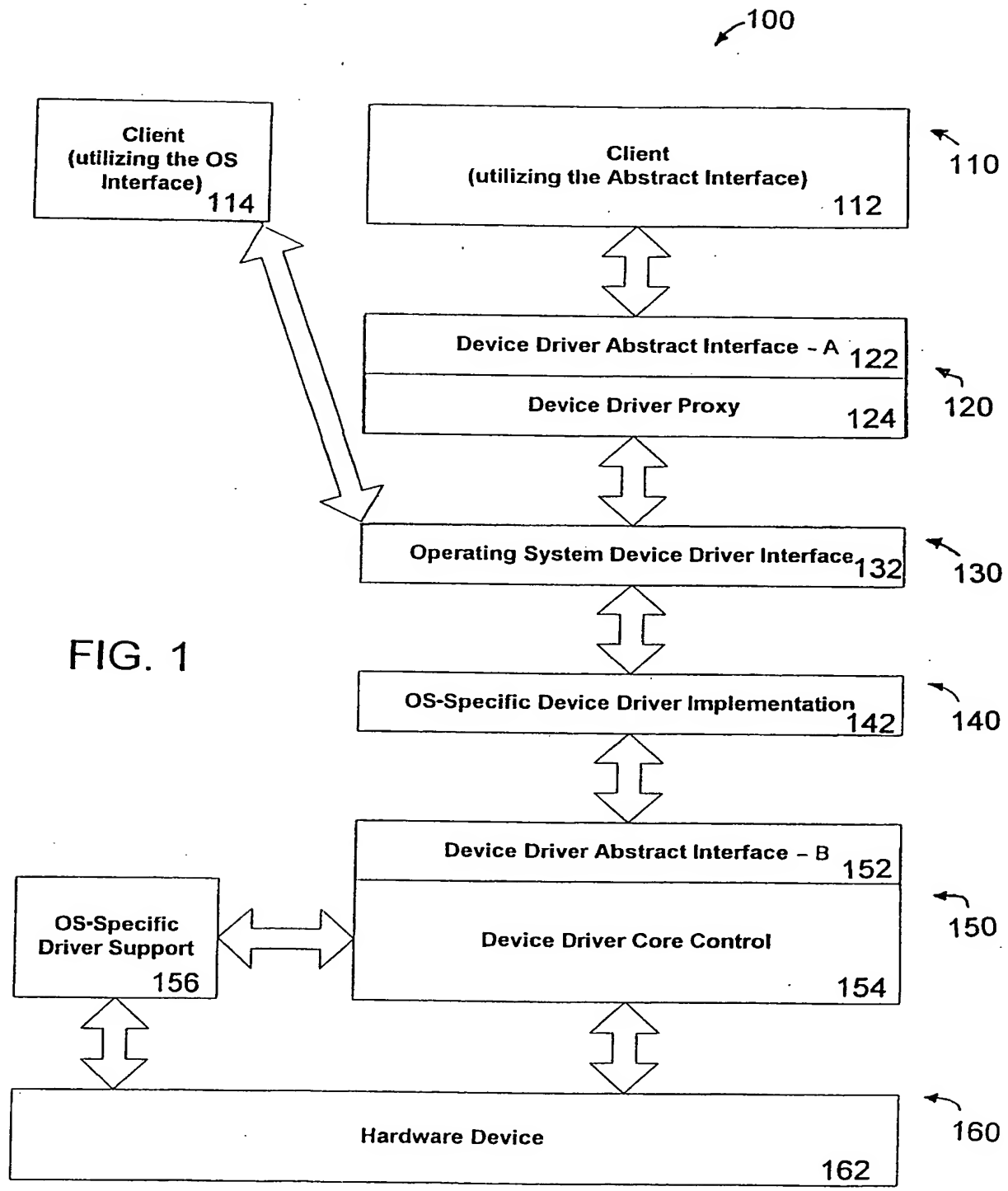


FIG. 1

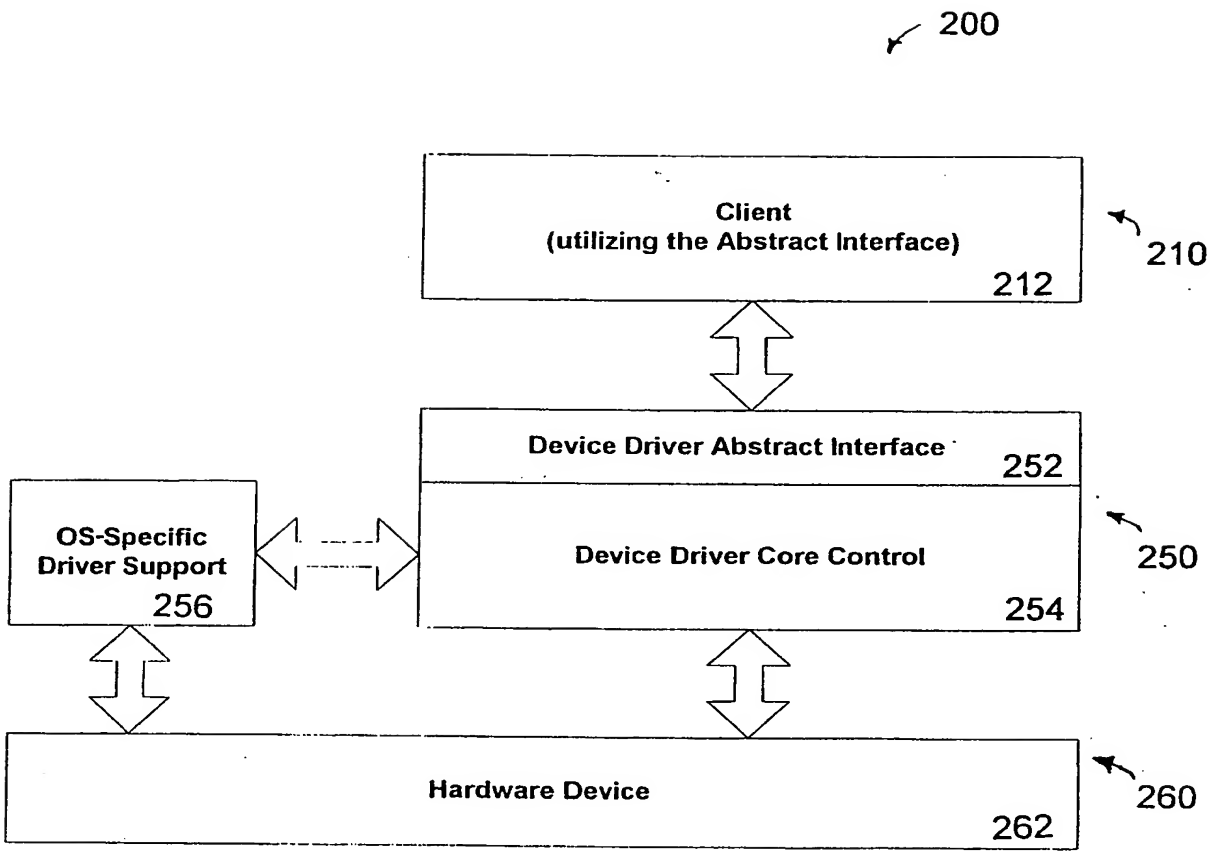

200
Client
(utilizing the Abstract Interface)
212
210
Device Driver Abstract Interface
252
OS-Specific
Driver Support
256
Device Driver Core Control
254
250
Hardware Device
262
260

FIG. 2

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 June 2001 (28.06.2001)

PCT

(10) International Publication Number
WO 01/046819 A3

(51) International Patent Classification[7]: G06F 13/10, H04N 5/00

(21) International Application Number: PCT/US00/42614

(22) International Filing Date: 6 December 2000 (06.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/468,047 21 December 1999 (21.12.1999) US

(71) Applicant *(for all designated States except US)*: **GENERAL INSTRUMENT CORPORATION** [US/US]; 101 Tournament Drive, Horsham, PA 19044 (US).

(72) Inventors; and

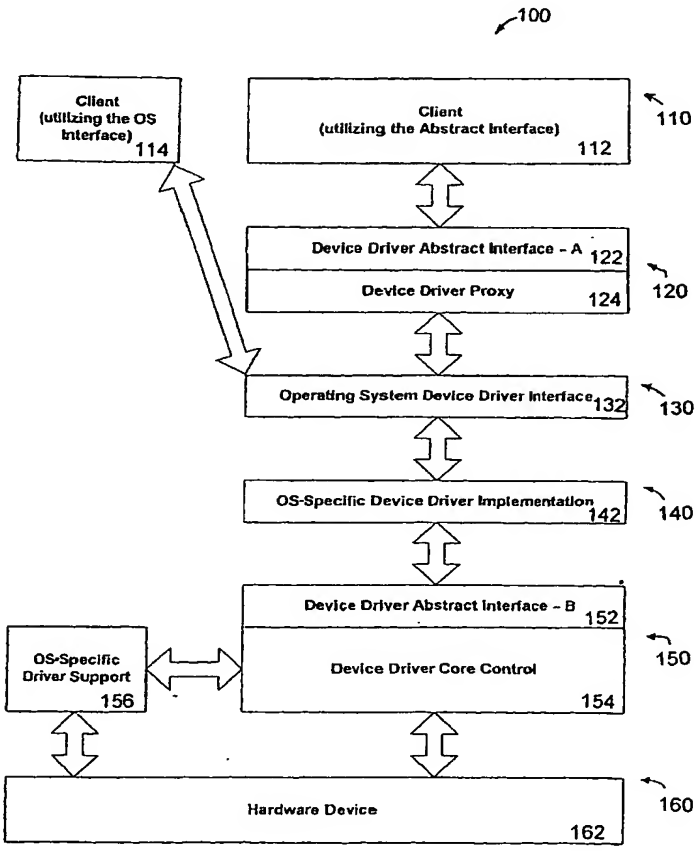
(75) Inventors/Applicants *(for US only)*: **GAZDA, Robert** [US/US]; 42 Le Forge Court, Wayne, PA 19087 (US). **DEL SORDO, Chris** [US/US]; 229 Heatherfield Drive, Souderton, PA 18964 (US). **PREZUHY, David, A.** [US/US]; 3398 Pin Oak Lane, Chalfont, PA 18914 (US). **BIRNBAUM, Jack, M.** [US/US]; 559 Nicole Drive, Southampton, PA 18966 (US).

(74) Agent: **LIPSITZ, Barry, B.**; Law Offices of Barry R. Lipsitz, 755 Main Street, Building No.8, Monroe, CT 06468 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: ABSTRACT DEVICE DRIVER MODEL FOR THE PORTABILITY OF DEVICE DRIVERS ACROSS DIFFERENT OPERATING SYSTEM PLATFORMS



(57) Abstract: An abstract interface model for device drivers in a set-top terminal, such as for use in a cable or satellite television subscriber network. In a layered software architecture, a device driver abstract interface model allows communication between a client (112, 212) and the device driver (154, 254) regardless of the operating system (OS) under which the device driver operates. In a first embodiment, the architecture includes a dedicated (OS-specific) device driver interface (132), and a proxy uses a first abstract interface (122) to convert interface service requests from a client (112) into OS-specific calls to the device driver (154). For a client that directly accesses a dedicated (OS-specific) device driver interface (132), a second abstract interface (152) is used to convert the interface service requests into OS-specific calls to the device driver (154). Direct access to the OS-specific device driver interface (132) is therefore maintained.

WO 01/046819 A3

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) **Date of publication of the international search report:**
28 November 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Inte nal Application No
PCT/US 00/42614

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F13/10 H04N5/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	RYAN S J: "SYNCHRONIZATION IN PORTABLE DEVICE DRIVERS" OPERATING SYSTEMS REVIEW (SIGOPS), ACM HEADQUARTER. NEW YORK, US, vol. 32, no. 4, October 1998 (1998-10), pages 62-69, XP000849896 the whole document --- -/--	1-15

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 May 2002	28/05/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Beltrán-Escavy, J

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	VAUGHAN P L ET AL: "Migrating from PVM to MPI.I. The Unify system" FRONTIERS OF MASSIVELY PARALLEL COMPUTATION, 1995. PROCEEDINGS. FRONTIERS '95., FIFTH SYMPOSIUM ON THE MCLEAN, VA, USA 6-9 FEB. 1995, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, 6 February 1995 (1995-02-06), pages 488-495, XP010130246 ISBN: 0-8186-6965-9 the whole document	1-15
A	WRIGHT M: "INTELLIGENT I/O: DOES I 2 O HOLD H 2 O ?" EDN ELECTRICAL DESIGN NEWS, CAHNERS PUBLISHING CO. NEWTON, MASSACHUSETTS, US, no. EUROPE, 2 March 1998 (1998-03-02), pages 56-60,62,64,, XP000779112 ISSN: 0012-7515 the whole document	1-15
A	GOTT R A: "INTELLIGENT I/O EASES SUBSYSTEM DEVELOPMENT" COMPUTER DESIGN, PENNWELL PUBL. LITTLETON, MASSACHUSETTS, US, vol. 37, no. 5, 1 May 1998 (1998-05-01), pages 106,108-110, XP000791246 ISSN: 0010-4566 the whole document	1-15
A	US 5 878 237 A (OLARIG SOMPONG P) 2 March 1999 (1999-03-02) abstract column 1, line 1 -column 13, line 20	1-15

INTERNATIONAL SEARCH REPORT

mation on patent family members

Inte ıal Application No
PCT/US 00/42614

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5878237 A	02-03-1999	NONE	